The Journal of Open Source Education

Practical machine learning with PyTorch

Jack Atkinson [1¶] **and Jim Denholm** [1]

1 Institute of Computing for Climate Science, University of Cambridge, UK ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Submitted: 28 October 2023
Published: unpublished

Summary

In the last decade machine learning (ML) and deep learning (DL)[1] have revolutionised many fields within science, industry, and beyond. Researchers across domains from the physical sciences to the digital humanities are increasingly looking to leverage these tools in their research. Many will be experts within their own domains, but will not have received any training in machine learning.

We have developed, and delivered, a set of materials entitled *Practical machine learning with PyTorch*, designed to teach participants how to *actually* write and run ML code in a *hands-on* fashion whilst also illustrating important design considerations.

Statement of need

With the explosion of ML and DL there have been several promising opportunities to apply these techniques in research. There are notable applications across many fields from the physical sciences (Carleo et al., 2019), climate science (Kashinath et al., 2021), to the digital humanities (Gefen et al., 2021).

Whilst there exist many examples of ML code online, it is often in the form of complete codes to be downloaded, read, and run by the user. These are often missing any discussion of theory, the development process, or alternative approaches beyond the scope of the specific example. In contrast, much theoretical ML material addresses high-level concepts without discussing coding considerations or details of how to actually use popular frameworks to implement the models.

Many know how ML works in an abstract sense, but will be unfamiliar with lower-level practicalities such as image transforms and other preprocessing techniques required to present data to neural networks. They can describe how something works, but would have no idea where to start if asked to do it. Such practical aspects are ideally learnt through trial-and-error and hands-on experience.

Many machine learning frameworks are accessed using a Python framework. One such commonly used framework is PyTorch (Paszke et al., 2019). Researchers are likely to have experience writing Python code, but not PyTorch.

Learning objectives

The key learning objective from this workshop could be simply summarised as:
"Provide participants with the ability to develop ML models in PyTorch".

However, there are a few subtleties that we wish to highlight. We go beyond the ability to blindly run downloaded code to:

[1]We will use the term ML when talking about both ML and DL in this article

The Journal of Open Source Education

- provide an understanding of the structure of a PyTorch model and ML pipeline,
- introduce the different functionalities PyTorch might provide,
- encourage good research software engineering (RSE) practice, and
- exercise careful consideration and understanding of data used for training ML models.

With regards to specific ML content we cover:

- using ML for both classification and regression,
- artificial neural networks (ANNs) and convolutional neural networks (CNNs), and
- treatment of both tabular and image data.

Teaching materials

All of the teaching materials for this course are available online in a GitHub repository. In addition we have a GitHub pages site as a central resource to point participants to.

Slides

We have produced two slide decks for the course, both available online and linked from both the repository and the GitHub pages site. The slides are written in Quarto (Allaire et al., 2022) markdown and rendered as reveal.js html. Source and instructions on how to render are included in the repository should others wish to tailor them to their specifications.

The first set of slides covers the machine learning content introducing deep learning and neural networks through the concept of optimisation and gradient descent which should be a familiar concept to participants. They then cover the concept of convolutional layers as a method to map and abstract image-like data for use in a neural network.

The second set of slides contains a discussion of where machine learning has been deployed in the field of climate science. This includes domain-specific concepts to be aware of in data-preparation and deployment.

To make the slides available online we use a GitHub action on the repository to render the slides and publish them to the GitHub pages site whenever there is a push to the main branch.

Exercises (jupyter notebooks)

The main material is composed of four jupyter notebooks, each containing a standalone exercise that takes participants through the process of developing and training an ML model, from data preparation and training to running inference. Each exercise is broken down into a number of subtasks (jupyter cells).

The code has been packaged using `pyproject.toml`. This means that installation for use in the workshop is simplified to cloning the material repository and running:

```
python -m pip install .
```

We advise users do this from within a virtual Python environment, instructions for which are provided under 'Installation and setup'. From there the jupyter notebook exercises are activated from the command line with `jupyter notebook`.

The first pair of exercises uses Palmer Penguins (Horst et al., 2020), a tabular dataset of penguin characteristics designed for exploration and visualisation. The source code associated with the project provides the scaffold to create a torch `Dataset` from this data. We do this to remove the burden from participants allowing them to focus on learning the key features of PyTorch in the early exercises. We review this code during the workshop to understand its functionality and how data can be prepared for use in training.

1) Penguin Species Classification

Classification of penguin species based on other physical characteristics.
This exercise takes participants through the process of writing an ANN. The tabular data from the *palmer penguins* dataset is read in and transformed using idiomatic PyTorch data-loading objects before creating dataloaders and introducing the concepts of training and validation splits. As part of this exercise we discuss how to prepare a dataset in terms of identifying unsuitable characteristics that could introduce bias, unintended behaviour, or spurious results in the learning process. We also introduce one-hot-encoding as a method to balance loss between different classes.
Data preparation is followed by creating a net from scratch, introducing loss functions and optimisers, and writing a training and validation loop. Finally, we proceed beyond simply training the model, completing the exercise by inspecting metrics, visualising results, and deploying the model to perform inference in a practical manner – a step that is often missing from ML tutorials.

2) Penguin Regression

Prediction of penguin mass (regression) based on other physical characteristics.
The second exercise is similar to the first, using an ANN to learn from *palmer penguins* data, but focusses on regression rather than classification. The procedure is largely the same, with a discussion around how the relevant features of the dataset are different to those selected in exercise 1. We highlight how appropriate choice of loss (objective) function allows us to leverage an identical architecture for a different applications – binary-cross-entropy for classification and mean-squared-error for regression. The TorchTools package (Denholm, 2023) is also introduced to simplify the process of creating neural nets.

3) MNIST Classification

Classifying handwritten digits from the MNIST database (LeCun, 1998) using a CNN.
MNIST digit classification is a popular choice for those learning ML as it provides a tangible objective. In this exercise we deal with image data, and how to represent them as a tensor, and cover various pre-processing techniques and transforms that may be applied. We also introduce torchvision, and the concepts of using public datasets from `torchvision.datasets` and pre-trained models from `torchvision.models`.

4) Ellipse Regression

Estimating the centroid of an ellipse (regression) from an image using a CNN.
The final exercise uses a custom dataset generated for this workshop. It consists of RGB images of ellipses along with coordinates of the centre and the major and minor radii. A similar process to all the other exercises is followed; preparing the data, adapting a pretrained model, training, and evaluating. This time there is less explicit guidance in the notebook as participants are familiar with the process and are becoming self-sufficient.

Throughout the notebooks we provide specific links to the PyTorch documentation where relevant. This is done to show participants where to find information to aid development and debugging, and where they can explore other options (optimisers, loss functions, transformations etc.) beyond those used in the course.

We also provide the notebooks as Google Colab instances allowing users to run the notebooks entirely from within their browser. This also enables the code to be run on a GPU (graphics processing unit) to speed up computation in the more complex exercises. This is particularly useful as the course is typically delivered to participants using laptops, most of which will not have a GPU. The Colab notebooks are stored in an adjacent branch of the repository, but can be launched through links in the README or website[2].

[2]A google account is required.

Solutions

Worked solutions to all of the exercises are provided in the form of completed notebooks including example output. These are available both in the repository and also as Colab instances.

Whilst we discuss RSE principles during the course and provide examples, there is often not time, nor is it conducive, to write docstrings and apply type hints to every function as we write them. The worked solutions are complete with docstrings (NumPy convention) and type-hints (checked by mypy). In a similar manner, though we emphasise the importance of code style and PEP8 during the course, we cannot guarantee that our, or the participants', code will be compliant. The worked solutions are linted using pylint and conform to the black code format, however, allowing introduction of these useful tools.

Content Delivery

The course has been designed to be very flexible in terms of delivery, allowing it to be adapted to and reused in various setups.

The main aspect we wish to emphasise in delivery is teaching via jupyter notebooks in a *"code-along"* fashion. This helps with engagement, participation, and understanding (Barba et al., 2019) and is essential, we feel, to having a long-lasting benefit. This approach slows those leading the course towards the rate at which the participants are working, and illustrates through errors (whether intentional or not!) that even experienced coders are human and make mistakes. Such errors can illustrate common pitfalls and provide an opportunity to include the teaching of debugging approaches. More generally this approach helps emphasise RSE principles, as participants can see the live application of these ideas in practise.

In terms of structure we suggest starting the lecture material on ANNs followed by the first pair of exercises before returning to the CNN lectures and then exercises 3 and 4. This allows the course to conveniently be broken into two parts (ANNs and CNNs) as, say, morning and afternoon or day 1 and day 2.

The lectures can vary in length depending on the prior experience of the participants and we encourage active participation and discussion. We suggest having a chalkboard on hand to expand on and illustrate concepts such as optimisation, activation functions, matrix algebra etc. Much like the code-along approach, this slows the lecturer down to the pace of those taking notes and allows for tailoring of the content to the participants.

Whilst the course can be delivered entirely as a code-along, we have also taught exercises 3 and 4 as a "lab", with participants working individually or in small groups supported by floating demonstrators. An advantage of this approach for the CNN exercises is that it allows participants to explore a variety of PyTorch's features, e.g. different image transformations, for themselves.

We also believe that there is sufficient guidance in the notebooks to follow the exercises alone, and we include a link to a recording of the first workshop. This is, however, no substitute to in-person delivery where participants can ask questions, and successive workshops are continually improving.

Teaching experience

This project was originally designed to be taught at two climate science summer schools. The first time delivered in two half-day workshops, the second as a single one-day workshop. No plan survives contact with the enemy, and we found is that it is not possible to complete all of the material in single day. We chose to focus on exercises 1 and 3, with exercises 2 and 4 being "homework".

Perhaps the most notable improvement following delivery was the addition of Colab instances of the notebooks. We found participants had often not completed the setup instructions in advance and subsequently experienced issues running on their local machines in the workshop. Problems were often specific to the individual and ate up a lot of time trying to understand polluted environments (xkcd 1987), unfamiliar IDEs and operating systems etc. to ensure everyone could participate. Participants who have not prepared and experience issues are now asked to activate the Colab notebooks, thereby not being left behind nor wasting the time of others.

Another useful lesson was that those with Apple Silicon machines can use the MPS backend to accelerate training, and without this the CNN exercises are prohibitively slow on these machines. As a result we added MPS detection to the notebooks alongside CUDA.

We encourage participants to feed experiences back into the project, either via a GitHub issue or pull request. This allows us to continually learn from delivery and improve the material for future participants, especially if making instructions clearer or providing solutions to previously unencountered problems.

Finally we observe that the lecture on domain-specific applications of ML was effective in tying the workshop together and encouraging participants to consider how they might utilise ML in their own work. This session was followed by good questions and discussion good and illustrates how to apply what has been learnt along with domain specific things to be aware of. We encourage anyone using this material to tailor this final set of slides to their own domain.

Acknowledgments

We thank anyone who has made a contribution to these materials, however small, assisted in code review for us, or helped as demonstrators on the course.

The Institute of Computing for Climate Science is funded by the Schmidt Futures Foundation.

References

Allaire, J. J., Teague, C., Scheidegger, C., Xie, Y., & Dervieux, C. (2022). *Quarto* (Version 1.2). https://doi.org/10.5281/zenodo.5960048

Barba, L. A., Barker, L. J., Blank, D. S., Brown, J., Downey, A. B., George, T., Heagy, L. J., Mandli, K. T., Moore, J. K., Lippert, D., & others. (2019). *Teaching and learning with jupyter*. https://jupyter4edu.github.io/jupyter-edu-book/

Carleo, G., Cirac, I., Cranmer, K., Daudet, L., Schuld, M., Tishby, N., Vogt-Maranto, L., & Zdeborová, L. (2019). Machine learning and the physical sciences. *Reviews of Modern Physics*, *91*(4). https://doi.org/10.1103/RevModPhys.91.045002

Denholm, J. (2023). *TorchTools*. https://github.com/jdenholm/TorchTools

Gefen, A., Saint-Raymond, L., & Venturini, T. (2021). AI for digital humanities and computational social sciences. *Reflections on Artificial Intelligence for Humanity*, 191–202. https://doi.org/10.1007/978-3-030-69128-8_12

Horst, A. M., Hill, A. P., & Gorman, K. B. (2020). *palmerpenguins: Palmer Archipelago (Antarctica) penguin data*. https://doi.org/10.5281/zenodo.3960218

Kashinath, K., Mustafa, M., Albert, A., Wu, J., Jiang, C., Esmaeilzadeh, S., Azizzadenesheli, K., Wang, R., Chattopadhyay, A., Singh, A., & others. (2021). Physics-informed machine learning: Case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A*, *379*, 20200093. https://doi.org/10.1098/rsta.2020.0093

LeCun, Y. (1998). *The MNIST database of handwritten digits*. http://yann.lecun.com/exdb/mnist/

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., & others. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, *32*.